Towards the Development of a Multi-Agent Cognitive Networking System for the Lunar Environment

Rachel Dudukovich
Cognitive Signal Processing Branch
NASA Glenn Research Center
Cleveland, Ohio
rachel.m.dudukovich@nasa.gov

Katherine Wagner
SCaN Intern
NASA Glenn Research Center
Cleveland, Ohio
kwagne27@uic.edu

Shilpa Kancharla
SCaN Intern
NASA Glenn Research Center
Cleveland, Ohio
skancha@ncsu.edu

Jason Fantl
SCaN Intern
NASA Glenn Research Center
Cleveland, Ohio
jasonemerald@gmail.com

Alex Fung
SCaN Intern
NASA Glenn Research Center
Cleveland, Ohio
alexfung@berkeley.edu

***Abstract*—This paper details the development of a multi-agent cognitive system intended to optimize networking performance in the lunar environment. NASA's current concept of the future of lunar communication, LunaNet [1], outlines a complex network of networks. Challenges such as scalability, interoperability and reliability must first be addressed to successfully fulfill this vision. Machine intelligence can greatly reduce the reliance on human operators and enable efficient operations for tasks such as scheduling and network management. The application of machine learning, artificial intelligence, and other automated decision-making techniques can be used to allow network nodes to intelligently sense and adapt to changes in the environment such as link disruptions, new nodes joining the network, and support for a diverse range of protocols. Cognitive networking seeks to evolve these technologies into an autonomous system with improved science data return, reliability, and scalability. In this paper, we study three main areas a means to further develop cognitive networking capabilities: networking and flight software development, analysis of wireless data for modeling and simulation, and development of algorithms for a multi-agent system.**

***Index Terms*—Cognitive Networking, Delay Tolerant Networking, Multi-Agent Reinforcement Learning**

I. Introduction

Future plans for the lunar communication architecture detail a wide variety of missions. Within the time frame of 2018-2028, over lunar 40 missions are planned among multiple space agencies, with vehicles including: lunar orbiters, surface mobile and stationary vehicles, lunar relays, Earth orbiting relays and Earth ground stations [2]. Space internetworking will be a key technology to develop, which must be capable of supporting Earth-to-Moon links, lunar crosslinks, and orbiting relays that may use RF or optical physical layers. The wide level of diversity among nodes makes interoperability a challenge within multiple levels of the network stack. In addition, the network may be highly dynamic, requiring scalability and expandability. It is for these reasons that reliance on human operators and predefined communication schedules will become increasingly impractical. Cognitive networking seeks to develop algorithms and protocols that will sense, decide and act autonomously according to changes in the network. This paper outlines the development of a cognitive networking system prototype to address the challenges of the future lunar network.

The work discussed in this paper is based on a student-led effort encompassing several key areas of the cognitive system including: embedded software development, delay tolerant networking, machine learning/artificial intelligence, network modeling and simulation. The networking concept builds upon the framework of delay tolerant networking (DTN). We model a realistic cognitive networking experiment with a commercial off-the-shelf (COTS) flight computer similar to hardware that has been used by previous NASA CubeSat experiments.

The prototype system builds upon the High-Rate Delay Tolerant Networking (HDTN) software developed by NASA Glenn Research Center [3],[4],[5]. The HDTN software has been released under the NASA Open Source Agreement and is available on GitHub [6]. Previous work has been focused on the development of store-and-forward, UDP, TCP, STCP and LTP convergence layers. A simple flow-based scheduler has been developed, which will be used to implement Contact Graph Routing (CGR) as well as additional algorithms which may be used to improve performance of routing and link selection. Section II of this paper, "Delay Tolerant Networking Prototype", discusses a basic implementation of the flight computer prototype and the DTN scenario.

To further develop upon the concept of link selection and routing, we analyze several real-world wireless data sets.

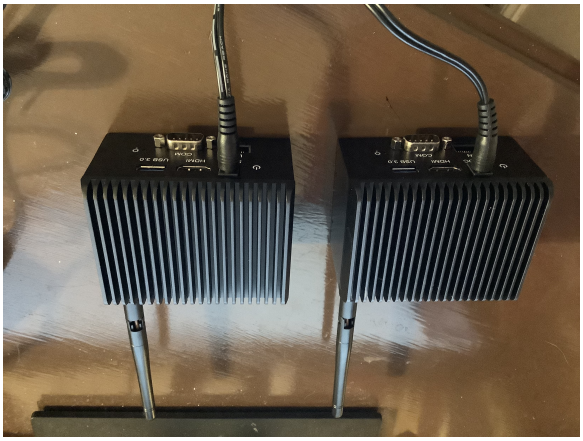

Fig. 1. Flight Computer Prototype

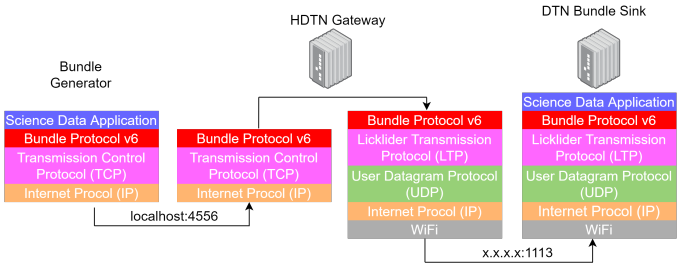

Fig. 2. Prototype Network Stack

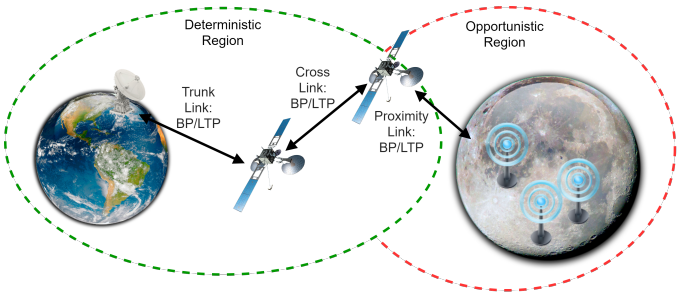

Fig. 3. Simple Lunar Architecture

In Section III, "Data Exploration and Analysis", we discuss metrics that may be used to assess link quality in a machine learning model, such as Received Signal Strength Indicator (RSSI). We develop a regression model which can predict link quality to enable informed routing decisions. We additionally discuss ways in which this data may be used to improve simulation capabilities.

Finally, we discuss the development of a multi-agent reinforcement learning (MARL) model. A MARL approach can be used to take into account the interactions of the various decision-making elements involved in link quality assessment, data selection, and routing, as well as how decisions of one node affect another. These interrelated decisions impact network congestion and remaining buffer/storage capacity in multiple areas of the network. We discuss multi-agent techniques which may apply to this problem as well as tools such as Ray and RLlib [7], which may be used for the simulation of the scenario.

II. Delay Tolerant Networking Prototype

Our prototype hardware is based on the Up Core gateway. The Up Core board consists of a 4-core Intel Atom x5-z8350 processor, 4 GB of RAM, and 32 GB Embedded MultiMediaCard (eMMC) storage. The gateway also provides 1 GbE LAN and WiFi 802.11 b/g/n. The prototype software is built on Ubuntu 20.04.2 LTS. Fig. 1 shows two of the gateways, each are 110x84x24 mm (LxWxH) in size. Similar, although not identical boards from Up have been used in CubeSat experiments.

The network stack implemented in the prototype system is shown in Fig. 2. Licklider Transmission Protocol in addition to Bundle Protocol is featured in the lunar architecture [2] to support long distance links with reliable transfer. Links from the moon directly to Earth may require this capability. Wireless LAN nodes on the surface may function within their own subnetwork, that will transmit data to a lunar relay or lunar gateway as shown in Fig.3.

In this scenario, we envision two classes of networks. The first is a Mobile Ad hoc Network (MANET) on the lunar surface consisting of both robotic nodes mobile nodes and stationary nodes. Network disruptions may occur as vehicles enter lunar craters or move out of range of other nodes. DTN protocols can be used to mitigate these disruptions and allow nodes to collect data for an extended period of time before needing to offload data to a lunar orbiter. This type of network may benefit from opportunistic styles of routing, in which a known network topology may be difficult to obtain. The second type of network consists of the lunar orbiters, relays and Earth ground stations. The connections in this type of network are well-defined, with known contact times and number of participating nodes. This network will also benefit from the use of Bundle Protocol and Licklider Transmission Protocol to account for periods of disconnection. However, this type of network is deterministic and is quite suited to Contact Graph Routing.

III. Data Exploration and Analysis

While the primary goal of this work is to investigate multi-agent reinforcement learning approaches which learn their environment in real-time (online learning), these methods may still require an initial model developed from a data set. Algorithms such as Deep Q-Learning and Advantage Actor-Critic both may start with an initial training phase which will require data. For this reason, this work conducted a preliminary data exploration phase in search of wireless data sets that can be used for building basic machine learning models and initial training data. In previous work [8], we discussed how link quality predictions could be used as an input into a deep reinforcement learning agent which could then decide the quantity of data fragments to send at a given time. This mechanism can also be used to select an output

link (or outduct) among multiple possible choices. To evaluate a wide range of available data sets, we considered several metrics such as Received Signal Strength Indicator (RSSI), data rate, and contact duration.

Two resources that were used for the data exploration are the Community Resource for Archiving Wireless Data at Dartmouth (CRAWDAD) [9] as well as a comprehensive link quality estimation survey [10]. The survey in [10] notes the challenge of finding suitable wireless data sets for machine learning. Many data sets have an insufficient number of samples, missing data, unlabeled data, or unclear formats. This is true even for terrestrial wireless networks but is even more of a challenge for space related networks such as DTNs, although there are a few DTN related data sets available on CRAWDAD. This preliminary study was based on terrestrial data sets due to these limitations, although the basic concepts and tool chain would apply to a variety of mobile networks.

A. ORBIT Data Set

Our initial data set consists of measurements from the Rutgers Open-Access Research Testbed for Next-Generation Wireless Networks (ORBIT) [11]. This data set includes the received signal strength indicator (RSSI) for each correctly received frame at the receiver node when various levels of noise are injected on the ORBIT testbed. The Rutgers ORBIT testbed data set is particularly of interest since the Cognitive Communication project at NASA Glenn is building an RF testbed similar to the ORBIT concept. The ORBIT testbed data can give insight into metrics and data set formats useful for machine learning.

The features used in this analysis include the following:

- Received: whether the signal was received or not (Boolean value)
- Error: indicates if an error has occurred while capturing the RSSI (Boolean value)
- Noise: amount of noise injected
- t_x: x-coordinate of the grid node that was configured as the transmitter (integer value)
- t_y: y-coordinate of the grid node that was configured as the transmitter (integer value)
- r_x: x-coordinate of the receiver node (integer value)
- r_y: y-coordinate of the receiver node (integer value)

The preprocessed data set resulted in 1,218,000 data points.

Several regression methods were trained on the ORBIT data set including: multiple linear regression, ridge regression (L2-norm), LASSO regression (L1-norm), random forest regression, Bayesian ridge regression, and finally XGBoost regression. To assess the performance of the regression models, the root mean squared error (RMSE) was be used. The RMSE measures the average magnitude of the error. Since the errors are squared before they are averaged, the RMSE gives a relatively high weight to large errors. In order to understand if large errors are present, the RMSE is particularly useful. A training, validation, and test split of the original dataset was created at 75%, 12.5%, and 12.5% respectively. Fig. 4 shows a comparison of the RMSE results for each regression method.

Fig. 4. Comparison the RMSE results for regression values among the training, validation, and test sets for the ORBIT data set.

Lastly, the optimal hyperparameters used for XGBoost regression were found using AWS SageMaker. These hyperparameters were used to inform the model created by hand as well. This approach of both "hand-done" and automated tuning ultimately yielded the best result.

B. DieselNet Data Set

While the ORBIT data set includes interesting characteristics such as varying noise levels injected into the system, the data set is based on stationary nodes that do not truly emulate a DTN environment. For this reason, we evaluated the DieselNet data set [12]. DieselNet is a network of 35 buses on the campus of University of Massachusetts, Amherst, which traveled planned routes every day throughout a 5-month period in the Spring 2006 semester. The buses were equipped with radio transmitters, and the trace set includes data for each one-way connection when the buses came into close enough proximity to transfer data. There were approximately 60,000 one-way contacts, each of the following format:

"Bus 3112 at 72.532745 42.393852 on route 1 in contact with bus 3114 at 72.532745 42.393852 on route 1 at time 418:27 for 45204688 bytes in 184792.0 ms."

DieselNet approximates a DTN environment well due to several factors. The link quality in DieselNet can be measured by throughput, or the amount of data transferred over a connection per unit time. For our lunar scenario, the bus mobility roughly simulates the movement of vehicles on the lunar surface. The regularity of bus routes could be envisioned as the movements of lunar rovers performing periodic maintenance activities. However, there are also weaknesses with this data set. Data collection was imperfect: the GPS coordinates were the same for both buses for each contact, the reported time duration was negative for a small fraction of the contacts, and some were missing data fields. After filtering these out, about 50,000 of the initial 60,000 contacts remained. However, many of these were "repeat" contacts – the same two buses phased in and out of connection up to roughly ten times. Furthermore, one-way contacts often failed to translate to two-way contacts: the unevenness of the histogram in Fig. 5 implies that the radios on some buses were systematically better at receiving or sending radio signals than others. Lastly, the documentation stated that the route information was invalid, so this was discarded. In addition, the concept of GPS does not directly

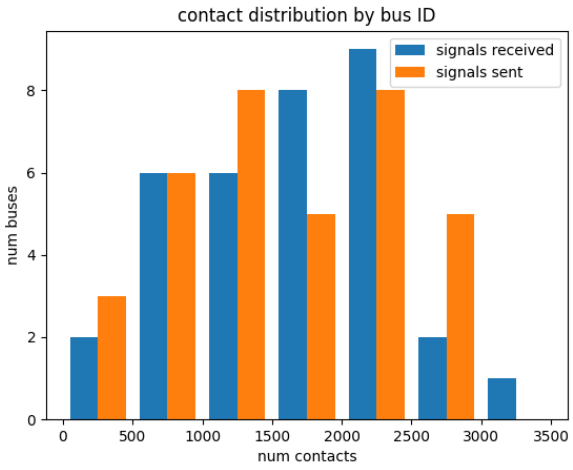

Fig. 5. Contact distribution of DieselNet buses

translate to the lunar environment. Fig. 5 shows the contact distribution of the DieselNet data set.

IV. Multi-Agent System

Effective CGR implementations have been explored using a variety of classical and machine learning algorithms with effective success [13]. However these well-tested approaches necessarily resolve decisions at the level of the immediate contact and potentially waste processing resources to re-appraise routing with changes in network state. These approaches also face realistic issues in accommodating partial access to and interruption of node and network state information. Hybrid scenarios as described in Section II may benefit from occasional opportunistic exploitation of resource availability [14], [15], [16].

A more comprehensive model of the CGR problem in space networks captures the context as a multi-agent system. Multi-agent techniques are especially adept at optimizing decision-making in complex, distributed environments [17]. For the CGR problem in particular a multi-agent framework acknowledges node interconnectivity and the progressive evolution of network state shaped by the interacting decisions of the node-agents [18].

In this scenario, each contact acts as a semi-independent collaborating decision-maker in a shared setting to produce a set of routing decisions that inform the decisions made by subsequent points of contact [17]. Using a reinforcement learning approach, the multiple agent nodes linked in the network are exposed to a series of training scenarios and learn to select progressively optimized routing schemes in response to a reward applied to the decisions made every time-step [19]. The objective is a networked system that makes more efficient networking decisions by acknowledging the action of each contact node [20].

Of particular interest for comparison are two chief MARL algorithms that have been widely explored in multi-agent learning contexts and have shown considerable promise in collaborative and distributed multi-agent settings, including in networking problems [21], [22]: Deep Q-learning and Advantage Actor-Critic.

In Deep Q-learning [23], a Q-value function is estimated in each time-step to refine the rational value assigned to each action available to the agent. Through training, this value function shapes an optimal process of decision making by each agent-node that is sensitive to the decisions made by the other agent nodes [19], [24].

Advantage Actor-Critic (A2C) [25] in contrast uses a centralized value function to "critique" and subsequently shape the particular utility assessment with which each individual node actor makes a decision. Replacing use of the Q-function, the Advantage function computes a relative value of each action as informed by the cumulative choices of all node-actors compared to an average action choice. Since the critic-assessed advantage function is updated more frequently than that of the actors, on each time-step the actor value function is purported to improve more rapidly with critic influence than without [26],[27].

There are several network-related tasks which a reinforcement learning agent may act upon:

- Routing and Link Selection - As stated above, a key area of focus has been the selection of an optimal end-to-end path in a multi-hop network [28]. The link quality estimates discussed in Section III can be used to select the most reliable physical connection among multiple choices (relay versus direct-to-earth or optical versus RF). A reinforcement learning agent can be used to take both link quality and network congestion into consideration and adjust accordingly.
- Data Selection - The criticality of data, remaining time-to-live, and size of the data when compared to remaining link capacity can serve as objectives in a multi-objective optimization [8].
- Parameter Tuning - LTP is a complex protocol with a variety of parameters. A reinforcement learning agent can be used to tune these parameters and improve performance. This is of particular concern if nodes in an opportunistic network implement neighbor discovery. Nodes can share initial connection information using beacons, and parameters can be further tuned by the agent.

A. Simulation

A concept of the simulation environment is shown in Fig. 6. Our initial prototype focused on routing and link selection, with the thought that the development of a basic framework could later be applied to other decisions described above (data selection and parameter tuning). The frequency at which messages reach their destinations was used as the objective metric for a central critic. Other metrics should be explored later since the current objective may encourage a sort of "message highway", where if a message is lucky enough to be on this highway it will reach its destination almost immediately, but if a message is unlucky then it may never reach its destination at all. This was not an issue in the simulated environment since

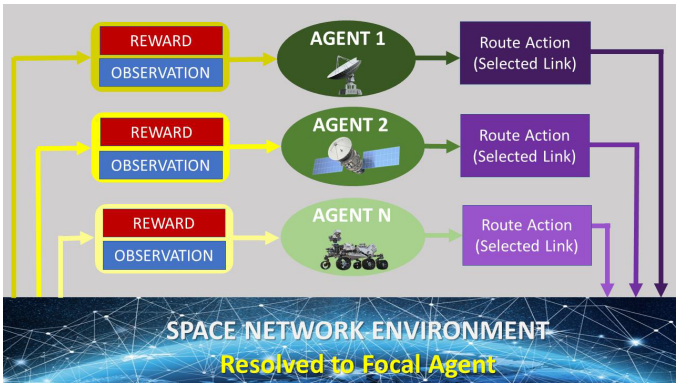

Fig. 6. Multi-Agent Environment Concept

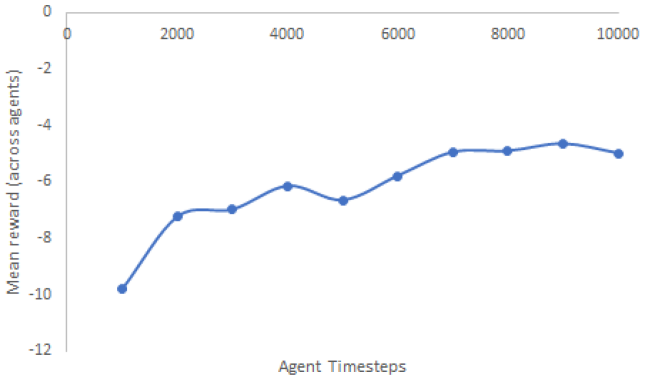

Fig. 7. Agents Learn to Reduce Latency

the connections were sparse enough that no "highway" could be established. If there is no centralized critic, then each agent must determine how successful a decision was based on local information. In this case, it may be best to measure throughput multiplied by how much closer a message is to its destination. The two perspectives of a centralized decision-maker versus a decentralized system will play an important role in further investigations.

In order to test different concepts and models, a machine learning framework was necessary to act as the interface between the custom DTN environment and different machine learning models. Ray [29] is an open source Python machine learning library focused on distributed learning. It provides an API for multi-agent environments as well as a number of well-known multi-agent models to train with. It also has the ability to train a number of models on a distributed system, which can improve training times.

The simulation environment models a node with several characteristics related to DTN. This model had a collection of agents, all of which held a queue of packets (a queue because the order of packets to send is not yet considered within the simplified environment). These packets only contain the destination they are trying to reach, other features such as packet size or time-to-live are future work. At each time step, a connection matrix is randomly generated that tells each agent what connections are available to other agents. Currently each agent will evaluate each link individually, due to the fact that in the future new agents could enter the network, creating a dynamic observation space. If each link is evaluated individually then the observation space is constant, which is necessary for most machine learning models. The agent evaluates each link and selects the best, sending its next packet if the agent is confident enough in the connection. This step of evaluation is where a machine learning model could be applied.

A preliminary learning implementation was developed using PyTorch to create a 3-layer Deep Q-Network. The agent's action was to select the best link with a reward based on $-1 \times (latency)$. The observations consist of the neighboring nodes that are available to send data to. The simulation consisted of 100 nodes, with 2 rollout workers per node. Rollout workers in Ray enable parallel exploration of the environment and multi-policy activation which can lead to faster convergence and more robust learning. The nodes exchanged 500 bundles over 2000 time steps per episode. Fig. 7 shows that the agents demonstrate learning within 1000 time steps. The agents' reward increases as they make better decisions and learned to reduce latency by approximately 40%. Single agent learners maintained 15-20 s delay at 2000 time steps.

V. Conclusion

Several areas were developed to envision a multi-agent system to enable improved network routing for complex environments such as the future lunar network. These include the developement of a DTN bundle agent, an exploration of available wireless data sets that can be used for modeling link quality as input to the multi-agent reinforcement learning model (MARL), the MARL agent and associate algorithms and an environment to train and test the agent. There is a lack of publicly available space-related data sets and so simulation alone may be an appropriate approach. However, the concepts developed with terrestrial data sets may still apply. The basic MARL agent and environment developed show that there is potential for improvement over single agent approaches.

Future work will be to integrate the real-world data sets into the Ray environment and eventually create a MARL module for the HDTN software. Additional work can be done to improve the fidelity of the simulation in terms of packet characteristics or integration with an orbital analysis program or network simulator.

References

[1] D. J. Israel et al. "LunaNet: a Flexible and Extensible Lunar Exploration Communications and Navigation Infrastructure". In: *2020 IEEE Aerospace Conference*. 2020, pp. 1–14. DOI: 10.1109/AERO47225.2020.9172509.

[2] Interagency Operations Advisory Group. *The Future Lunar Communications Architecture*. Tech. rep. Report of the Interagency Operations Advisory Group Lunar Communications Architecture Working Group, 2019.

[3] A. Hylton et al. "Rising above the cloud: Toward high-rate delay-tolerant networking in low earth orbit". In: *Advances in Communications Satellite Systems. Proceedings of the 37th International Communications Satellite Systems Conference (ICSSC-2019)*. 2019, pp. 1–17. DOI: 10.1049/cp.2019.1216.

[4] A. Hylton and D. Raible. "High Data Rate Architecture (HiDRA)". In: *Ka and Broadband Communications Conference 2016*. 2016.

[5] R. Dudukovich et al. "A distributed approach to high-rate delay tolerant networking within a virtualized environment". In: *IEEE Cognitive Communications for Aerospace Applications Workshop*. 2021.

[6] *High-Rate Delay Tolerant Networking*. https://github.com/nasa/HDTN.

[7] Eric Liang et al. *RLlib: Abstractions for Distributed Reinforcement Learning*. 2018. arXiv: 1712.09381 [cs.AI].

[8] R. Dudukovich et al. "Microservice Architecture for Cognitive Networks". In: *Proceeding of the 2020 IEEE International Conference on Wireless for Space and Extreme Environments*. 2020.

[9] David Kotz et al. *CRAWDAD dataset dartmouth/campus (v. 2009-09-09)*. Downloaded from https://crawdad.org/dartmouth/campus/20090909. Sept. 2009. DOI: 10.15783/C7F59T.

[10] Gregor Cerar et al. "Machine Learning for Wireless Link Quality Estimation: A Survey". In: *IEEE Communications Surveys Tutorials* 23.2 (2021), pp. 696–728. ISSN: 2373-745X. DOI: 10.1109/comst.2021.3053615. URL: http://dx.doi.org/10.1109/COMST.2021.3053615.

[11] D. Raychaudhuri et al. "Overview of the ORBIT radio grid testbed for evaluation of next-generation wireless network protocols". In: *IEEE Wireless Communications and Networking Conference, 2005*. Vol. 3. 2005, 1664–1669 Vol. 3. DOI: 10.1109/WCNC.2005.1424763.

[12] John Burgess et al. *CRAWDAD dataset umass/diesel (v. 2008-09-14)*. Downloaded from https://crawdad.org/umass/diesel/20080914. Sept. 2008.

[13] Giuseppe Araniti et al. "Contact graph routing in DTN space networks: overview, enhancements and performance". In: *IEEE Communications Magazine* 53.3 (2015), pp. 38–46. DOI: 10.1109/MCOM.2015.7060480.

[14] S. Burleigh et al. "Toward a unified routing framework for delay-tolerant networking". In: *2016 IEEE International Conference on Wireless for Space and Extreme Environments (WiSEE)*. 2016, pp. 82–86. DOI: 10.1109/WiSEE.2016.7877309.

[15] Carlo Caini and Rosario Firrincieli. "Application of Contact Graph Routing to LEO satellite DTN communications". In: *2012 IEEE International Conference on Communications (ICC)*. 2012, pp. 3301–3305. DOI: 10.1109/ICC.2012.6363686.

[16] Rachel Dudukovich and Christos Papachristou. "Delay Tolerant Network Routing as a Machine Learning Classification Problem". In: *2018 NASA/ESA Conference on Adaptive Hardware and Systems (AHS)*. 2018, pp. 96–103. DOI: 10.1109/AHS.2018.8541460.

[17] Maria Amélia Lopes Silva et al. "A reinforcement learning-based multi-agent framework applied for solving routing and scheduling problems". In: *Expert Systems with Applications* 131 (2019), pp. 148–171. ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2019.04.056. URL: https://www.sciencedirect.com/science/article/pii/S0957417419302866.

[18] Edward Birrane, Scott Burleigh, and Niels Kasch. "Analysis of the contact graph routing algorithm: Bounding interplanetary paths". In: *Acta Astronautica* 75 (2012), pp. 108–119. ISSN: 0094-5765. DOI: https://doi.org/10.1016/j.actaastro.2012.02.004. URL: https://www.sciencedirect.com/science/article/pii/S0094576512000288.

[19] Jayesh K. Gupta, Maxim Egorov, and Mykel Kochenderfer. "Cooperative Multi-agent Control Using Deep Reinforcement Learning". In: *Autonomous Agents and Multiagent Systems*. Ed. by Gita Sukthankar and Juan A. Rodriguez-Aguilar. Cham: Springer International Publishing, 2017, pp. 66–83. ISBN: 978-3-319-71682-4.

[20] Ryan Lowe et al. "Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments". In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. NIPS'17. Long Beach, California, USA: Curran Associates Inc., 2017, pp. 6382–6393. ISBN: 9781510860964.

[21] Kaiqing Zhang, Zhuoran Yang, and Tamer Başar. *Multi-Agent Reinforcement Learning: A Selective Overview of Theories and Algorithms*. 2021. arXiv: 1911.10635 [cs.LG].

[22] Dmitry Mukhutdinov et al. "Multi-agent deep learning for simultaneous optimization for time and energy in distributed routing system". In: *Future Generation Computer Systems* 94 (2019), pp. 587–600. ISSN: 0167-739X. DOI: https://doi.org/10.1016/j.future.2018.12.037. URL: https://www.sciencedirect.com/science/article/pii/S0167739X18309087.

[23] V. Mnih et al. "Human-level control through deep reinforcement learning". In: *Nature* 518 (2015), pp. 529–533.

[24] Xinge Li et al. "A Multi-Agent Reinforcement Learning Routing Protocol for Underwater Optical Sensor Networks". In: *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*. 2019, pp. 1–7. DOI: 10.1109/ICC.2019.8761441.

[25] Volodymyr Mnih et al. *Asynchronous Methods for Deep Reinforcement Learning*. 2016. arXiv: 1602.01783 [cs.LG].

[26] Ryan Lowe et al. *Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments*. 2020. arXiv: 1706.02275 [cs.LG].

[27] Yan Zhang and Michael M. Zavlanos. *Distributed off-Policy Actor-Critic Reinforcement Learning with Policy Consensus*. 2019. arXiv: 1903.09255 [cs.LG].

[28] R. Dudukovich and A. Hylton. "A Machine Learning Concept for DTN Routing". In: *Proceeding of the 2017 IEEE International Conference on Wireless for Space and Extreme Environments*. Montreal, Quebec, 2017.

[29] Philipp Moritz et al. "Ray: A Distributed Framework for Emerging AI Applications". In: *CoRR* abs/1712.05889 (2017). arXiv: 1712.05889. URL: http://arxiv.org/abs/1712.05889.